50
185 pts.
IVA incluido

RUN

Enciclopedia Práctica del Spectrum

Nueva Lente/Ingelek

LOS MENSAJES DEL SPECTRUM

Cuando ejecutamos un programa en nuestro SPECTRUM, siempre aparece un mensaje, o bien de finalización o bien de error por haber cometido una equivocación en la programación. Estos mensajes nos dicen dónde está el fallo y tenemos una pista para solucionarlo.

Todo mensaje que emite el ordenador se divide en tres partes.

0 OK, 120.2

La primera de ellas es el primer carácter que aparece (en este ejemplo el «0»), que puede ser un número o una letra, y nos sirve para encontrar en el manual la explicación de dicho mensaje. La segunda es el mensaje en sí (el «OK» del ejemplo), que nos explica lo que ha ocurrido. La última siempre son dos cifras separadas por dos puntos (:). La primera de ellas es el número de línea y la segunda es en qué comando de dicha línea se ha detenido la ejecución (en este caso en la línea 120, comando número 2).

Ahora vamos a estudiarlos uno por uno y, así podremos saber más rápidamente cómo localizar los fallos y aplicar el remedio apropiado.

El ordenador imprime siempre unos mensajes para saber las incidencias que ocurran durante la ejecución del programa.

0 OK

Nos indica que la ejecución del programa ha sido satisfactoria, sin haber encontrado la máquina ningún error.

1 NEXT without FOR

Hallado **NEXT** sin existir **FOR**.

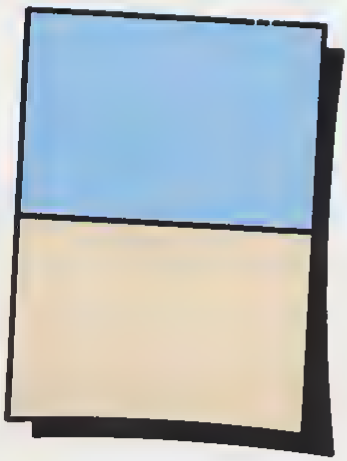
Estamos utilizando un bucle **FOR** ... **NEXT** y hemos olvidado escribir precisamente la sentencia **FOR ... TO**. El ordenador, sin embargo, sí ha encontrado en la ejecución del programa un **NEXT** x, sin estar previamente el **FOR** x=... TO...

2 Variable not found

El ordenador ha encontrado una variable no declarada previamente.

Es un error muy común en aquellos que antes de manejar el SPECTRUM han trabajado con otros ordenadores. Se produce cuando pretendemos trabajar con una variable sin antes haberla dado

Para evitar el mensaje **2 Variable not found** *hemos de realizar una asignación antes de utilizar la variable*



¡!

El mensaje **R Tape loading error** aparece cuando al efectuar **LOAD** o **MERGE** no es posible cargar, o al ejecutar **VERIFY** no se ha realizado una correcta verificación.

*

Si no hemos introducido la cinta en el ZX MICRODRIVE, o queremos trabajar con uno no existente, el ordenador imprimirá el mensaje **Microdrive not present**

¡!

Al operar o trabajar con una variable que no tenga un valor previamente asignado se produce el mensaje **2 Variable not found**. Se evita declarándola previamente, por ejemplo **LET x=0**

*

Si en una sentencia **INPUT** pulsamos **STOP**, o en una **INPUT LINE** pulsamos **CAPS SHIFT+6** se produce el mensaje de aviso **H STOP in INPUT**.

un valor inicial (lo que se llama declarar una variable). Así se puede dar en **PRINT x** o **LET x=x+1** si a la variable "x" no se le ha asignado un valor previo con **LET (LET x=0)**, **READ**, **INPUT**, **FOR** o **DIM** (en el caso que estemos trabajando con matrices y no se haya dimensionado antes con este comando).

3 Subscript wrong

Hemos asignado un subíndice erróneo en variables que los utilicen.
Interviene siempre una sentencia **DIM**. Ocurre cuando nos referimos a la tabla con un subíndice mayor que la dimensión que hicimos de ella. Así, si hemos efectuado **DIM x(90)** el error aparecerá al ejecutar **LET x(91)=...** También aparece cuando se ha definido una matriz de dimensión cero: **DIM x(0)**. Ver mensaje B.

4 Out of memory

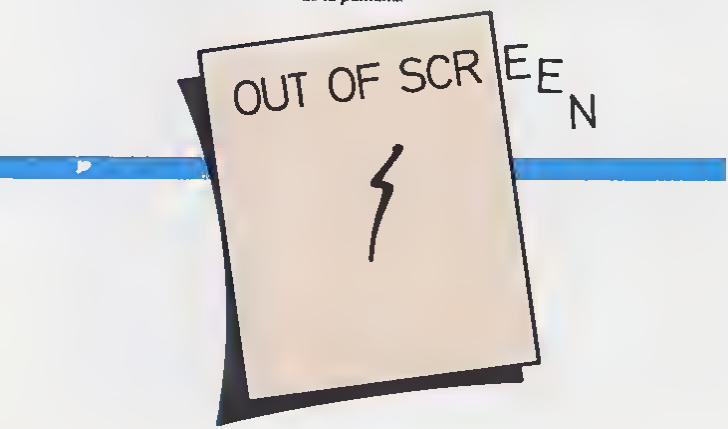
Se ha sobrepasado la capacidad de memoria de la máquina.
Nos avisa de la imposibilidad de realizar la operación pertinente al no tener suficiente capacidad el ordenador. Ocurre normalmente con **DIM**, al dimensionarla demasiado grande, con LOAD, al cargar un programa o una matriz, o MERGE, dado que la suma de los dos programas, el que se encuentra en memoria más el que se está cargando, sobrepasa la memoria libre.

Cuando pretendemos rebasar la capacidad de memoria, el ordenador nos avisa con **4 Out of memory**.

5 Out of screen

Hemos pretendido imprimir texto fuera de la pantalla.
Ocurre con **INPUT** cuando llevamos escritas 23 líneas. También aparece con **PRINT AT 22,...** puesto que pretendemos imprimir en una línea no existente (el SPECTRUM sólo dispone de 21 líneas de pantalla en donde podemos imprimir normalmente).

5 Out of screen *aparece cuando intentamos imprimir fuera de la pantalla.*



6 Number too big

Ha aparecido un número demasiado grande.
Esta situación se puede dar con cualquier comando que realice alguna operación aritmética. Se sobrepasa la capacidad del ordenador para operar porque el número tiene un valor muy alto y la máquina no puede seguir. Si no existe ningún error en el programa es necesario, para evitar la paralización del mismo, situar una sentencia **IF** y cuando se llegue a cierto valor desviar su ejecución.

7 RETURN without GO SUB

El ordenador ha localizado un **RETURN** sin existir un **GO SUB**.
Este error se da frecuentemente cuando se producen varias llamadas a subrutinas y, por tanto, se sitúan varios **GO SUB** y **RETURN**. Puede ocurrir que se haya emplazado algún **RETURN** sobrante o falten **GO SUB**.

8 End of file

Se ha llegado al final del fichero.
Sólo se produce cuando estemos trabajando con MICRODRIVE, e indica que hemos pretendido leer más datos de los existentes en un fichero.

9 STOP statement

En el programa hay una sentencia **STOP.**
El ordenador, en la ejecución de un programa, se ha encontrado dicha sentencia y, por tanto, se detiene, avisandonos

A Invalid argument

Tenemos un argumento no válido
Ocurre sobre todo con las funciones matemáticas **ASN, ACS, LN, SOR, ^**, al intentar operar con numeros negativos. Esto es debido, en todos los casos menos **SDR**, a que el ordenador realiza todas las operaciones con el sistema logaritmico y éste no admite dichos números. Con **SDR** ya sabemos que no podemos operar normalmente con números negativos (es necesario recurrir a los números complejos y crearnos un programa para ello)

B Integer out of range

Tenemos un número fuera del margen permitido
Se produce con todos los comandos, que no sean funciones matemáticas, cuyos argumentos sean números (**RUN, POKE, GD TO, LIST, DIM...**). Es debido a que dicho argumento es incorrecto para lo que se le requiere. Así con **DIM** (ver error 3) ocurre cuando se dimensiona una matriz con números negativos o mayor que 65535 (si es menor de esta cifra, 65534, aparece el mensaje 4)

C Nonsense in BASIC

No es entendido un comando
Cuando **VAL** y **VAL$** utilizan argumentos incorrectos. Por ejemplo **VAL 10**. Si el zumbido de exceso de memoria ha sonado, independientemente del motivo, el ordenador ha podido producir «cambios extraños» en el listado del programa, siendo imposible su ejecución normal, apareciendo este mensaje. La única solución es corregirlo a mano y esperar que no haya afectado a algo más y sea necesario empezar de nuevo

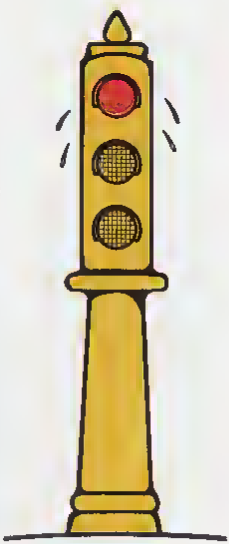
Cuando queremos seguir leyendo información en el MICRODRIVE y no hay más datos, aparece el mensaje **8 End of file**

Cuando por el resultado de las operaciones, una variable tome un valor demasiado alto aparecera el mensaje **6 Number too big**

D BREAK-CONT repeats

Se ha detenido voluntariamente el programa.
Cuando se pulsa BREAK (CAPS SHIFT+SPACE) en alguna operación con el casete (**LOAD, MERGE, SAVE, VERIFY**), en alguna salida por impresora o al bloquear el *scrool?* de un listado. En la primera situación, si se hace **CONTINUE** el comando se volverá a ejecutar de nuevo desde el principio. Ver el mensaje L.

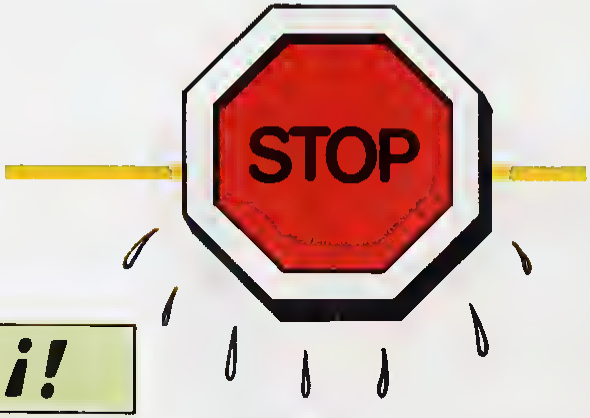
Al pulsar la tecla **BREAK** *provocamos la detencion del programa*



E Out of DATA

Nos hemos quedado sin datos
Hemos utilizado la sentencia **READ** más veces de lo que podemos, y tras leer todos los datos del comando **DATA** se ha continuado leyendo, al no existir más datos, se ha producido la situación errónea. Comprobando el listado, debemos contar cuantas veces se utiliza **READ** y cuantos datos hay en **DATA** para saber donde se ha producido el error

F Invalid file name



Cuando el ordenador se encuentra un STOP, *la ejecución del programa se detiene apareciendo el mensaje* 9 STOP statement.

¡!

Todo mensaje del ordenador se divide en 3 partes: la primera es un número o una letra, que nos sirve para buscar en el manual de qué mensaje se trata, la segunda es el propio mensaje y la tercera son dos cifras que nos indican el número de línea y el comando de dicha línea, respectivamente, donde se cometió el error.

*

C Nonsense in BASIC, además de con **VAL** y con **VAL$**, puede aparecer cuando, tras sonar el zumbido de exceso de memoria, se ha producido una alteración en el listado del programa realizada por el propio ordenador.

El nombre del archivo (programa o *array*) es incorrecto.
Al grabar en el casete con **SAVE** se utilizó un nombre incorrecto o con más de 10 caracteres o con ninguno.

G No room for line

No tenemos más espacio para líneas de programa.
El ordenador nos avisa, por falta de memoria, que no podemos introducir más líneas. En cualquier forma, el SPECTRUM no permite un número de orden mayor que 9999: si ejecutamos **GO TO 10000**, o similar, nos aparecerá el mensaje de error B.

H STOP in INPUT

Programa interrumpido en un **INPUT**.
Hemos provocado la detención del programa en una sentencia **INPUT** o **INPUT LINE**. En el primer caso, cuando se introduce un dato numérico, se detiene con **STOP (SYMBOL SHIFT+A)**, si es alfanumérico hay que borrar previamente la comilla de la izquierda y luego escribir **STOP**. En el segundo caso, detenemos el programa con **CAPS SHIFT+6**. En todas las situaciones aparece el mensaje H.

I FOR without NEXT

Hay un **FOR** sin el **NEXT** correspondiente.

K INVALID COLOUR

Si pretendemos utilizar un color no correcto, el programa se detiene apareciendo **K Invalid colour**

Cuando se nos olvida ubicar un NEXT, *al no tener bucle final, el ordenador nos avisa con el mensaje* 1

Observe que se encuentra muy relacionado con el mensaje 1. En este caso, al contrario, se nos ha olvidado situar el final del bucle con **NEXT x**, suponiendo que el comienzo es **FOR x=...** Estos errores suelen aparecer en los bucles anidados, y lo mejor es seguir fielmente el programa sobre un papel para comprobar dónde se encuentra el error.

J Invalid I/O device

No es válido el dispositivo de entrada salida.
Este mensaje se da con operaciones con el MICRODRIVE únicamente.

K Invalid colour

El color utilizado no es válido.
El parametro de un comando gráfico, ya sea **OVER, BRIGHT, FLASH, INVERSE, INK, PA-**

PER o **BOROER**, ha sido incorrecto. No debemos olvidarnos que los tres primeros comandos sólo pueden tomar los valores 0 y 1, y el resto de 0 a 7 para los colores, 8 para caracteres transparentes y 9 para el modo de contraste (estos dos últimos números no los maneja **BORDER**)
Este mensaje es muy común también cuando estamos trabajando en BASIC y la memoria se ha visto afectada, ya sea accidental o intencionadamente.

L BREAK into Program

Programa interrumpido
En la ejecución de un programa hemos presionado la tecla **BREAK (CAPS SHIFT+SPACE)**. Muy relacionado con D, pero aquí no se trata de operaciones con el casete.

M RAMTOP no good

RAMTOP no válido
Al efectuar un **CLEAR x**, el parámetro **x**, tiene un valor demasiado pequeño o demasiado grande.

N Statement lost

Se ha perdido una sentencia
Ocurre cuando se produce un salto a una línea que no existe con **CONTINUE**, **RETURN**, **NEXT**. En el primer caso, se produce después de **BREAK**, y es aconsejable volver a ejecutar el programa con **GO TO**.

O Invalid Stream

Al trabajar con el MICRODRIVE, tenemos una serie no valida.

P FN without OEF

Hallado **FN** sin haber encontrado previamente **DEF FN**.
Cuando trabajamos con una función es necesario que vayan emparejados dos comandos **OEF**

Si aparece **D BREAK-cont repeats** *o* **L BREAK into Program** *es que hemos detenido voluntariamente el programa apretando* **BREAK**.

FN y **FN**. Este mensaje de error indica que el ordenador no ha encontrado la definición de la función cuando se le ha pedido trabajar con ella.

Q Parameter error

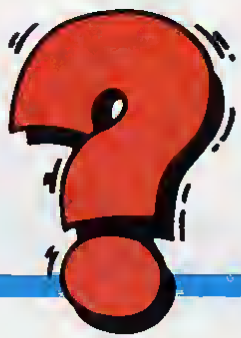
Hemos introducido un parámetro erróneo
Ocurre con el comando **FN**. Al operar se ha encontrado con un parámetro no correcto, por ejemplo, una cadena alfanumérica en el lugar de numérica, o si en nuestra función hay potenciaciones, ha aparecido un número negativo al operar.

R Tape loading error

Error de carga en el casete
Cuando hemos procedido a cargar en el ordenador, con **LOAD** o **MERGE**, o a verificar la graba-

El ordenador no «sabe» operar siempre con números negativos

$$(-2)^2$$



¡!

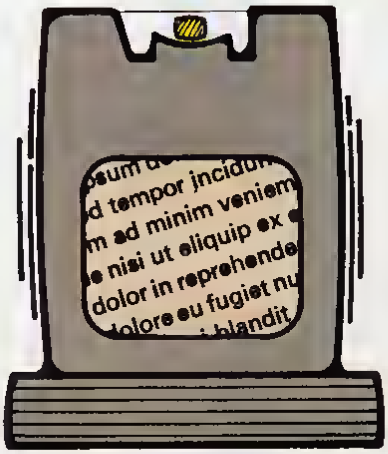
Cuando leemos (**READ**) más datos de lo que se encuentra en **DATA** aparece el mensaje **E Out of DATA**.

*

Cuando los argumentos de algunas funciones son incorrectos aparece el mensaje **A Invalid argument**.

*

Si pretendemos acceder a un dato en una tabla con un subíndice incorrecto (negativo o más alto que la dimensión de la propia tabla) aparece el mensaje **3 Subscript wrong**.



MICRODRIVE FULL

Cuando operamos con un cartucho de MICRODRIVE en el que no cabe más información aparece el mensaje **Microdrive full.**

MENSAJES CON EL ZX-MICRODRIVE

Microdrive not present

No hay MICRODRIVE
Este mensaje nos puede aparecer en dos situaciones. La primera es cuando pretendemos trabajar sin tener la cinta del MICRODRIVE introducida en el aparato. Si hemos indicado que queremos trabajar con un ZX MICRODRIVE que no tenemos conectado (por ejemplo, tenemos un solo aparato y señalamos que queremos trabajar con el segundo), también aparece este mensaje.

Microdrive full

El MICRODRIVE está completo.
No podemos cargar más información en la cinta de MICRODRIVE al no disponer de espacio. Podemos saber continuamente el espacio libre si realizamos de una manera periódica **CAT**. Aparece un listado de los programas y ficheros en la cinta y los Kbytes libres.

File not found

Fichero no hallado
Al ejecutar **LOAD** o **MERGE** el programa indicado no ha sido encontrado. Realizando **CAT** sabremos si de verdad falta o hemos cometido un error al escribirlo.

Verification has failed

Verificación fallada
Se ha producido un error al verificar (**VERIFY**) un programa, posiblemente por un fallo en la grabación (**SAVE**).

ción (**VERIFY**) ha ocurrido un error que hace imposible dicha carga. En la primera situación hay que repetir la operación, para comprobar si vuelve a aparecer el mensaje. Si se estaba ejecutando **VERIFY** se debe volver a grabar con **SAVE**. Este tipo de error puede surgir por muy diferentes motivos, desde que el aparato de grabación se ha detenido accidentalmente, hasta un fallo en él o en la cinta. En este caso, se puede saber mirando si está doblada en algún punto, o si mirando el cuentavueltas del aparato, el mensaje R aparece siempre en el mismo número, se impone una sustitución de ésta.

¡!

Cuando realizamos varias llamadas a subrutinas es posible que aparezca el mensaje **7 RETURN without GO SUB** al colocar más comandos **RETURN** que **GO SUB**.

*

Con **0 OK** el ordenador nos indica que el programa se ha ejecutado correctamente hasta el final.

*

Cuando ejecutamos un programa en nuestro SPECTRUM siempre aparece un mensaje o bien de fin o bien de error, por haber cometido una equivocación en la programación.

C Nonsense in BASIC *aparece cuando un comando no es entendido por el ordenador*

MANIPULANDO BITS

Hasta el presente capítulo, todas las instrucciones discutidas mantenían el objetivo común de efectuar una determinada operación sobre un octeto o grupo de ellos, pero no podíamos actuar sobre un bit concreto del cual, por cualquier razón, fuera preciso escrutar o modificar su contenido. Precisamente, una de las mejoras que el Z 80 incorpora frente a su predecesor el 8080, la constituye el numeroso grupo de instrucciones encargadas de gestionar información a nivel de bit individual (312 nuevas instrucciones).

TEST, SET Y RESET

En la página 849 bajo la denominación «MANIPULACION DE BITS» están recogidas todas las instrucciones encargadas de examinar (TEST) el contenido del bit señalado dentro de un registro o posición de memoria determinada, colocarlo a 1 (SET) fuera cual fuera su estado, o fijarlo a 0 (RESET).

Los mnemónicos asociados con estas operaciones son BIT b,s SET b,s y RES b,s respectivamente, donde s es el contenido del registro o posición de memoria (8 bits) a la que acceder y b, el lugar ocupado por el bit individual, sobre el cual vamos a actuar.

Recordemos el convenio seguido en toda la literatura informática, según el cual dentro de un octeto, los bits que lo constituyen van numerados de 0 a 7 comenzando por el de la derecha.

Supongamos, por ejemplo, que el acumulador contiene en el momento previo a efectuar una de estas instrucciones 10110011. Comprobemos el efecto que sobre el valor almacenado en A obran la ejecución sucesiva de algunas de estas órdenes:

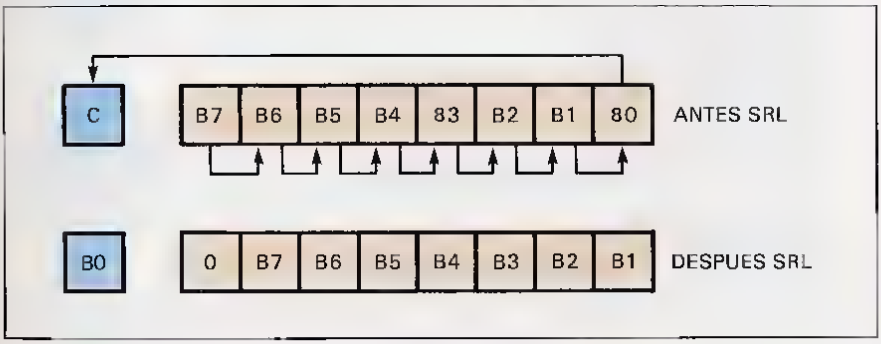
INSTRUCCION	ACUMULADOR
SET 2,A	10110111
SET 4,A	10110111
RES 7,A	00110111
RES 6,A	00110111

Fácilmente se comprueba que tanto las instrucciones SET como las de RESET son independientes del valor original del bit afectado, es decir, fijan su contenido a **1** (SET) o a 0 (RES) sin atender a su estado anterior.

Otra circunstancia a tener en cuenta es que implementarlas, no provoca cambio en el registro de indicadores F. Sin embargo, diferente es lo que ocurre cuando nuestro objetivo es escrutar (TEST) un bit específico.

Supongamos ahora, que el acumulador mantiene el valor obtenido tras la instruccion RES 6,A. Efectuemos algunas operaciones de TEST sobre algunos bits del byte almacenado en A:

INSTRUCCION	ACUMULADOR	INDICADOR DE CERO
BIT 0,A	00110111	0
BIT 6 A	00110111	1



¡!

Todas las instrucciones de rotación y desplazamiento afectan a los indicadores S, Z, P/V y C, a excepción de las cuatro anteriores.

*

Al ejecutar RLD o RRD rotamos los dígitos decimales de un numero expresado en formato BCD.

*

Las instrucciones de desplazamiento tienen especial utilidad en la construcción de rutinas para la multiplicación y división multibyte.

La primera consecuencia que extraemos es que tras una instrucción BIT, el contenido del registro o posición de memoria implicada no varía. En nuestro ejemplo, el acumulador no ha experimentado cambio alguno.

La segunda es que el resultado se pone de manifiesto en función de la bandera de cero del registro F, es decir, si el bit es 0, el indicador Z contendrá 1, y 0 en caso contrario.

Estas circunstancias son, en particular, muy útiles cuando tratemos de determinar el valor de un bit concreto sin necesidad de modificar el contenido de A. Por ejemplo, las dos parejas de instrucciones siguientes conducen al mismo objetivo: determinar si el bit B6 del byte almacenado en el registro B es 1:

```
LD A,B          LD A,B
AND 40h         BIT 6 A
```

La primera secuencia, se vale del byte 40 como máscara para detectar si efectivamente el bit considerado es 1, modificándose para ello el contenido de A. La segunda realiza el mismo trabajo, pero A no se ve alterado. Ambas precisan 5 bytes para ser implementadas, y según las circunstancias nos decidiremos por una u otra técnica.

```
CONVERSOR BCD-ASCII

23296           LD A,02         ;pantalla superior
23298           CALL CHAN-OPEN  ;abre canal 2
23301           LD HL,9C40      ;origen de los números BCD
23304           LD C,cantidad   ;¿cuantos van a ser convertidos?
23306   BCD     LD B,02         ;cada número BCD tiene dos
                                ;dígitos
23308   LOOP    PUSH BC         ;guarda BC
23309           PUSH HL         ;guarda HL
23310           XOR A           ;borra el acumulador
23311           RLD             ;rotación decimal a la izquierda
23313           ADD A,30        ;sitúa en el nibble de mayor
                                ;orden 3
23315           RST 10          ;imprime carácter ASCII
23316           POP HL          ;recupera HL
23317           POP BC          ;recupera BC
23318           DJNZ LOOP       ;efectúa el bucle dos veces (dos
                                ;dígitos)
23320           INC HL          ;toma el siguiente BCD
23321           DEC C           ;queda uno menos que procesar
23322           JP NZ BCD       ;¿queda alguno por procesar?
23324           RET             ;vuelve al BASIC.
```

ROTACIONES

En la página 653 están recogidas todas las instrucciones de rotación y desplazamiento. Centrándonos en las de rotación, las cuatro primeras

MNEMONICO	CODIGO MAQUINA	REGISTRO F (7 6 5 4 3 2 1 0 / S Z H P/V N C)	Nº BYTES	CICLOS MAQ	CICLOS RELOJ	NOTAS
BIT b,r	1 1 0 0 1 0 1 1 / 0 1 x b x x r x	x ↕ 1 x 0 •	2	2	8	
BIT b (IX+d)	1 1 0 1 1 1 0 1 / 1 1 0 0 1 0 1 1 / x d x / 0 1 x b x 1 1 0	x ↕ 1 x 0 •	4	5	20	
BIT b (IY+d)	1 1 1 1 1 1 0 1 / 1 1 0 0 1 0 1 1 / x d x / 0 1 x b x 1 1 0	x ↕ 1 x 0 •	4	5	20	
SET b,S	1 1	• • • • • • • •				
RES b,S	1 0	• • • • • • • •				

r	
B	000
C	001
D	010
E	011
H	100
L	101
A	111
(HL)	110

S
r
(IX+d)
(IY+d)

NOTA: Las instrucciones SET y RES siguen idéntica decodificación a BIT, pero sustituyendo los bits B7 y B6 por las parejas 11 y 10 respectivamente.

líneas contienen los códigos asociados con estas operaciones

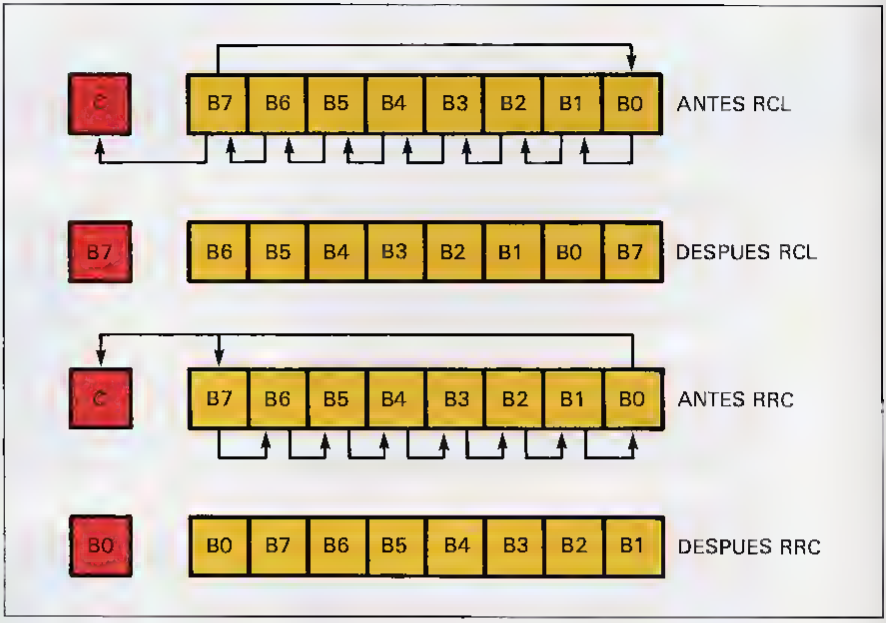
El primer grupo de instrucciones es el de rotaciones circulares hacia la izquierda (RLC, *Rotate Left*

MNEMONICO	CODIGO MAQUINA	REGISTRO F 7 S	6 Z	5	4 H	3	2 P/V	1 N	0 C	Nº BYTES	CICLOS MAQ.	CICLOS RELOJ	NOTAS
RLCA	0 0 0 0 0 1 1 1	•	•		0		•	0	↕	1	1	4	
RLA	0 0 0 1 0 1 1 1	•	•		0		•	0	↕	1	1	4	
RRCA	0 0 0 0 1 1 1 1	•	•		0		•	0	↕	1	1	4	
RRA	0 0 0 1 1 1 1 1	•	•		0		•	0	↕	1	1	4	
RLCr	1 1 0 0 1 0 1 1 0 0 0 0 0 x r x	↕	↕		0		P	0	↕	2	2	8	
RLC(IX+d)	1 1 0 1 1 1 0 1 1 1 0 0 1 0 1 1 x d x 0 0 0 0 0 1 1 0	↕	↕		0		P	0	↕	4	6	23	
RLC(IY+d)	1 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1 x d x 0 0 0 0 0 1 1 0	↕	↕		0		P	0	↕	4	6	23	
RL s	0 1 0	↕	↕		0		P	0	↕				
RRC s	0 0 1	↕	↕		0		P	0	↕				
RR s	0 1 1	↕	↕		0		P	0	↕				
SLA s	1 0 0	↕	↕		0		P	0	↕				
SRA s	1 0 1	↕	↕		0		P	0	↕				
SRL s	1 1 1	↕	↕		0		P	0	↕				
RLD	1 1 1 0 1 1 0 1 0 1 1 0 1 1 1 1	↕	↕		0		P	0	•	2	5	18	
RRD	1 1 1 0 1 1 0 1 0 1 1 0 1 1 1 1	↕	↕		0		P	0	•	2	5	18	

s	c
B	000
C	001
D	010
E	011
H	100
L	101
A	111
(HL)	110

S
r
(IX+d)
(IY+d)

NOTA. Las instrucciones donde en su código máquina sólo se dan tres bits siguen idéntica decodificación que los anteriores



¡!

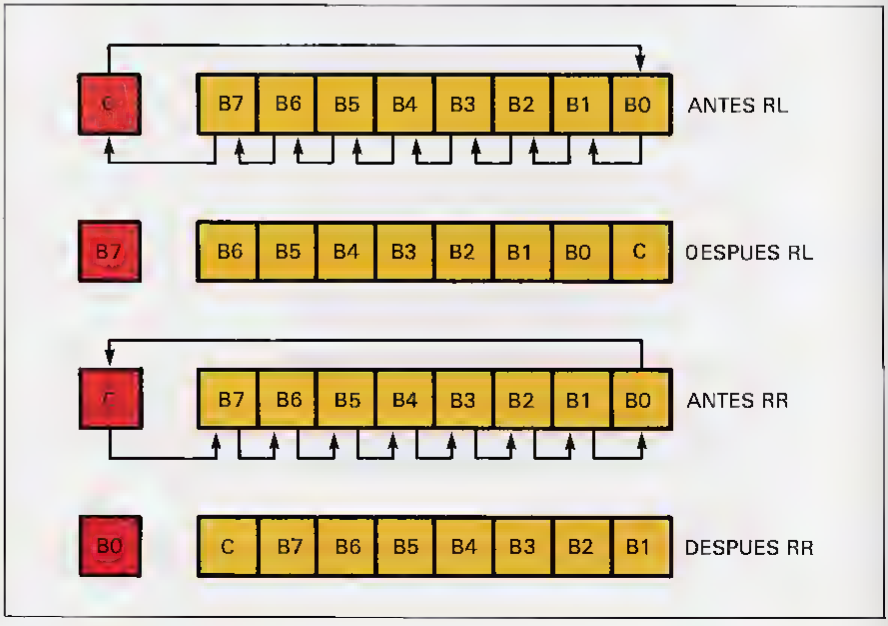
La instrucción de rotación derecha RR permite mediante aplicaciones sucesivas a un determinado octeto seleccionar el acceso a una secuencia de subrutinas, previamente escogida por nosotros

*

Los grupos de manipulación de bit, rotación y desplazamiento constituyen un nuevo bloque de instrucciones solamente contemplado en el Z 80, no así en su predecesor el 8080

Circular). Su mecanismo queda descrito en el diagrama de la figura, es decir, todos los bits del octeto considerado giran una posición hacia la izquierda, y el contenido del último se duplica en el lugar dejado vacante por primero y en el indicador de arrastre C.

La rotación derecha circular (RRC, *Rotate Right Circular)* es, en esencia, idéntica a la anterior,



pero ahora el giro se produce en sentido contrario, es decir, hacia la derecha y como el bit sobrante sería el B0, éste pasa a ocupar el lugar B7 y el de la bandera de arrastre
Para averiguar, por ejemplo, el bit de mayor orden que no es cero dentro de un determinado octeto, se emplean con mucha frecuencia las instrucciones de rotación izquierda (RL, *Rotate Left)*
Se trata de un giro completo de los bits del octeto especificado al que añadimos un noveno bit el indicador de acarreo
Puesto que B7 es transferido al indicador C cada vez que implementamos una instruccion de rotación izquierda, bastará escrutar el contenido de éste, hasta que finalmente almacene un 1
La rotación derecha (RR, *Rotate Right)* sigue el mismo principio de efectuar el giro de los nueve bits obtenidos al añadir el bit C al octeto sobre el cual operamos, aunque por supuesto, el desplazamiento se realiza en sentido contrario a las agujas del reloj
Su aplicación resulta inmediata en problemas en los que sólo se necesite acceder a determinadas subrutinas dentro de un bloque de éstas, y en un orden prefijado, de tal forma que no siempre se ejecuten todas, sino tan sólo las que necesitemos
En el diagrama hemos representado un ejemplo de como se implementaria esta tecnica manejando una instrucción de rotación derecha Observeremos que es posible fijar 256 combinaciones diferentes según nuestras necesidades

Al final de la tabla de la página 653 se encuentran cuatro instrucciones redundantes con las anteriores y que tan sólo actúan sobre el contenido del acumulador Se trata de RLCA, RRCA, RLA y RRA
Aun a pesar de poder resultar repetitivas, los fabricantes del Z 80 prefirieron mantenerlas para conservar la compatibilidad con el microprocesador 8080 y bajo determinadas condiciones nosotros podemos emplearlas con cierta ventaja sobre sus equivalentes, pues tan solo precisan de un byte para ser implementadas
Estas cuatro instrucciones sólo afectan al indicador de errastre C, a diferencia de todas las anteriores que además modifican el contenido de los de signo, paridad/sobrepasamiento, y cero Por tanto, cuando tras una operación de rotación solamente sea necesario tener presente el acarreo conseguiremos ahorrar tiempo de proceso y espacio de memoria implementando estas últimas.

DESPLAZAMIENTOS

Las instrucciones de desplazamiento constituyen un arma de inestimable valor cuando se trate de

¡!

Las instrucciones RLCA RRCA RLA y RRA solamente pueden trabajar con el contenido del acumulador y además, afectan exclusivamente al indicador de arrastre C

*

La instruccion TEST escruta un bit especifico dentro de un octeto y revela su estado activando o no la bandera de cero Z

*

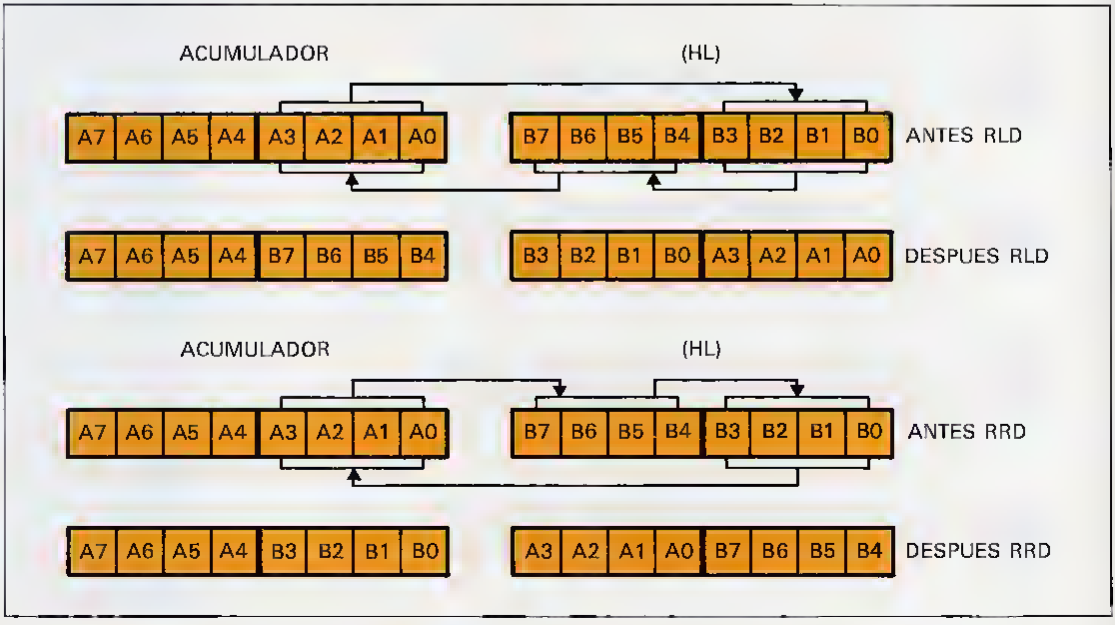
Las instrucciones SET y RES modifican el contenido del octeto implicado en ella, mientras que al ejecutar TEST este se mantiene inalterado

efectuar rutinas que resuelvan las operaciones de multiplicación y división de varios bytes
En su mecanismo interviene de manera protagonista el indicador de arrastre C, aunque tras procesarlas, las banderas Z, S y P/V serán afectadas según el resultado obtenido, al igual que ocurría con las instrucciones de rotación.
En los gráficos correspondientes se representa de forma esquemática su funcionamiento. Observemos que la instrucción de desplazamiento aritmético a la izquierda SLA *(Shift Left Arithmetic)*, provoca que el valor almacenado en el octeto afectado sea multiplicado por dos, proporcionando una señal de sobrepasamiento (se activa el indicador de arrastre), cuando el resultado sea mayor o igual de 256
Por el contrario, la instrucción SRA *(Shift Right Arithmetic*, desplazamiento aritmético a la derecha) divide el contenido del registro dado entre dos, situándose el resto de esta división (el microprocesador supone que los números están expresados en complemento a dos) en el indicador C
Finalmente, SRL *(Shift Right Logic*, desplazamiento lógico hacia la derecha), efectúa la división por dos del valor almacenado en el registro implicado por la instrucción, pero ahora considerando que está expresado en formato binario
Resulta útil considerar el uso conjunto de las instrucciones de rotación y las de desplazamiento al objeto de conseguir trasladar a la izquierda o derecha un grupo de varios bytes. Por ejemplo, la siguiente secuencia desplaza a la izquierda bit a bit el contenido del par HL, colocando en el lugar vacante un 0, lo cual equivale a la multiplicación por 2 del valor almacenado en dicho par.

```
SLA L
RL H
```

ROTACIONES EN BCD

Para terminar con el grupo de instrucciones de rotación y desplazamiento nos queda comentar dos de ellas, las cuales sirviéndose de direccionamiento indirecto (efectúan su trabajo sobre el octeto señalado por el par HL), ejecutan rotaciones a la izquierda o derecha de grupos de 4 bits (nibbles).
Al igual que antes, para no perder la información sobrante al ejecutar el giro (rotábamos bit a bit y el saliente pasaba al indicador C), el nibble saliente, ahora se recoge en los 4 bits de menor peso del acumulador A mientras que los previamente contenidos allí quedan almacenados en el nibble vacío del octeto considerado. En la figura se representa el proceso seguido



Por estas razones, las instrucciones RLD *(Rotate Left Decimal,* rotación decimal hacia la izquierda) y RRD *(Rotate Right Decimal,* rotación decimal hacia la derecha) son de especial utilidad cuando estemos manejando números expresados en formato binario codificado en decimal (BCD), pues efectúan una rotación entre los dígitos decimales almacenados en la posición de memoria indicada por HL y el acumulador.

En el próximo capítulo serán tratadas las instrucciones de entrada/salida de datos entre el microprocesador y periféricos.

Asimismo, abordaremos un tema de especial interés: las interrupciones.

TIRA PARED

¡!

Las letras que aparecen subrayadas en el listado deben introducirse en el modo gráfico (**CAPS SHIFT+9** O **GRAPHICS**).

*

Los literales **INV** y **TRUE**, subrayados y entre corchetes, corresponden a los caracteres de control **INVERSE VIDEO** y **TRUE VIDEO**, respectivamente.

*

Para la adopción del programa sólo tenemos que emplear, como de costumbre, el comando **SAVE "TIRA PARED"**, o bien **SAVE "TIRA PARED" LINE 1**, para que se efectue el proceso de autoejecución.

Nos encontramos ante uno de los grandes clásicos entre los programas de rapidez de reflejos. TIRA PARED. Esta versión del juego, aunque sólo haciendo uso del BASIC, consigue una velocidad auténticamente extraordinaria.

EL PROGRAMA

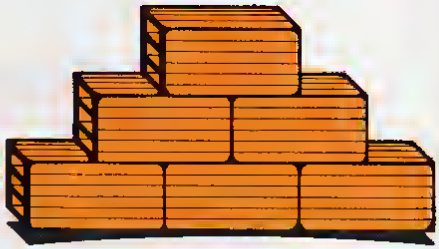
La finalidad del juego es muy sencilla, y seguramente conocida por la mayoría de nosotros. Aun así, vamos a recordar brevemente cuales son las reglas de nuestro programa TIRA PARED.

Inicialmente en la pantalla aparecen 12 filas de ladrillos simulando un muro. La pared está compuesta por un total de 198 ladrillos que debemos destruir en su totalidad. Para ello, disponemos de una bola desenladrilladora y de una lanzadera de dichas bolas.

La lanzadera actúa a modo de «cama elástica» en donde la bola botará para iniciar la destrucción del muro. Nosotros podemos controlar la lanzadera haciendo uso de las teclas «Z» y «X». La pulsación de la tecla «Z», desplazará a la lanzadera hacia la izquierda, mientras que la «X» lo hará hacia la derecha.

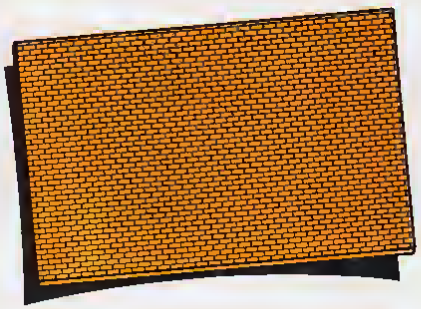
Siempre debemos ir al encuentro de la bola antiladrillo, puesto que de lo contrario, si el proyectil toca el suelo lo perderíamos. El ordenador posee una pantalla de captación de datos en la cual nos da a elegir el número de vidas que deseamos para una partida. El máximo de oportunidades no puede exceder de 9 y el mínimo no ha de bajar de 3.

Por último, si no logramos nuestro objetivo (derribar el muro), nuestro Spectrum, al finalizar la partida o lo que es lo mismo, al agotar el número de vidas nos dirá cuantos ladrillos nos han faltado para la destrucción total de la pared. Si por el contrario, logramos esa destrucción total de la muralla, el Spectrum nos felicitará, invitándonos a una nueva partida.



Al finalizar la partida, o lo que es lo mismo, al agotar el número de vidas, el Spectrum nos dirá cuantos ladrillos nos han faltado para conseguir nuestro objetivo.

El muro de TIRA PARED está formado por un total de 198 ladrillos, que es necesario destruir en su totalidad.



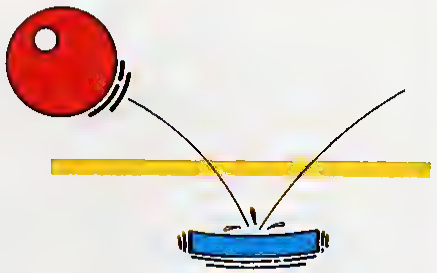
ADOPCION DEL PROGRAMA

Al constar simplemente de un soporte BASIC, la

adopción del programa es extraordinariamente sencilla; sólo tenemos que emplear como de costumbre, el comando **SAVE "TIRA PARED"**, o bien **SAVE "TIRA PARED" LINE 1**, para que se efectúe el proceso de autoejecución
En cuanto a las letras que aparecen subrayadas ya habremos adivinado que debe introducirse en el modo gráfico, y al igual que en otros listados, los literales **INV** y **TRUE**, subrayadas y entre corchetes, corresponden a los caracteres de control **INVERSE VIDEO** y **TRUE VIDEO**, respectivamente

```
  1 REM **********************
  2 REM * J.M.MAYORAL SERRANO *
  3 REM **********************
  4 REM * TIRA PAREO (C) 1985 *
  5 REM **********************
  8 POKE 23658,8
 10 GO SUB 940
 15 IF NOT PEEK 32767 THEN  GO SUB 1000
 20 LET TT=0: LET T=0: LET P=1: BORDER 0: INK 0: PAP
ER 7: CLS
 30 FOR M=1 TO 12 STEP 2
 40 FOR N=0 TO 30 STEP 2
 50 PRINT AT M+3,N; INK 7; PAPER 2;"AB",AT M+4,N;"BA

 60 NEXT N: NEXT M
 70 LET U=0: LET V=0
 80 LET A=14: LET T=0: LET W=0
 90 FOR R=1 TO TOPE
100 RANDOMIZE : LET M=15: LET N=INT (RND*3)+13
110 LET G=530: LET P=0: LET A=13
120 PRINT AT 21,0;"                EFG"
130 GO SUB G: PRINT AT U,V;" "
140 PRINT  INK 4;AT M,N;"C": LET U=M: LET V=N
150 IF M=20 THEN  PRINT  INK 4;AT M,N;"C"
160 LET K$=INKEY$
170 IF K$="Z" THEN  GO SUB 620
180 IF K$="X" THEN  GO SUB 660
190 GO TO 130
200 IF M>20 THEN  GO TO 680
210 IF M<20 THEN  GO TO 260
220 IF T>=198 THEN  GO TO 930
230 LET P=0: LET W=0: IF N=A+1 OR N=A+2 THEN  LET G=
300: GO TO 300
240 IF N=A THEN  LET G=500: GO TO 510
250 IF N=A-1 THEN  LET G=350: GO TO 350
260 IF N>30 THEN  LET G=400: GO TO 400
270 LET M=M+1: LET N=N+1
280 LET C=ATTR (M,N): IF C<>56 THEN  GO SUB 710: IF
P=0 OR W=1 THEN  LET G=300
290 RETURN
300 IF N>30 THEN  LET G=350: GO TO 350
310 IF M<1 THEN  LET W=1: LET G=200, GO TO 200
320 LET M=M-1: LET N=N+1
330 LET C=ATTR (M,N): IF C<>56 THEN  GO SUB 710: LET
P=1: LET G=200+150*W
340 RETURN
350 IF M<1 THEN  LET W=1: LET G=400: GO TO 400
360 IF N<1 THEN  LET G=300: GO TO 300
370 LET M=M-1: LET N=N-1
380 LET C=ATTR (M,N): IF C<>56 THEN  GO SUB 710: LET
P=1: LET G=400-100*W
390 RETURN
400 IF M>20 THEN  GO TO 680
410 IF M<20 THEN  GO TO 460
420 IF T>=198 THEN  GO TO 930
430 LET P=0: LET W=0: IF N=A+3 THEN  LET G=300: GO T
O 300
440 IF N=A+2 THEN  LET G=500: GO TO 500
450 IF N=A OR N=A+1 THEN  LET G=350: GO TO 350
460 IF N<1 THEN  LET G=200: GO TO 200
470 LET M=M+1  LET N=N-1
480 LET C=ATTR (M,N): IF C<>56 THEN  GO SUB 710. IF
P=0 OR W=1 THEN  LET G=350
490 RETURN
500 IF M<1 THEN  LET G=530: GO TO 590
510 LET M=M-1: LET C=ATTR (M,N): IF C<>56 THEN  GO S
UB 710: LET G=530
520 RETURN
530 IF M>20 THEN  GO TO 680
540 IF M<20 THEN  GO TO 590
550 IF T>=198 THEN  GO TO 930
560 LET P=0: LET W=0: IF N=A+2 THEN  LET G=300: GO T
O 300
570 IF N=A+1 THEN  LET G=300+(50*(1NT (RND*2))). GO
TO G
580 IF N=A THEN  LET G=350  GO TO 350
590 LET M=M+1: RETURN
600 IF A<1 THEN  RETURN
610 LET A=A-1: PRINT AT 21,A;"EFG ": RETURN
620 IF A<2 THEN  GO TO 680
630 LET A=A-2: PRINT AT 21,A;"EFG  ": RETURN
640 IF A>28 THEN  RETURN
```



La lanzadera actúa a modo de cama elástica, en la cual la bola rebotará para iniciar la destrucción del muro.

```
650 LET A=A+1: PRINT AT 21,A-1," EFG": RETURN
660 IF A>27 THEN  GO TO 640
670 LET A=A+2: PRINT AT 21,A-2;"  EFG": RETURN
680 BEEP .5,40: BEEP 1,10
690 NEXT R
700 GO TO 780
710 LET T=T+1
720 LET B=ABS (M-N)
730 LET Y=INT (B/2)*2
740 BEEP .01,40
750 IF B=Y AND N<31 THEN  PRINT AT M,N+1;" "
760 IF B<>Y AND N>0 THEN  PRINT AT M,N-1;" "
770 RETURN
780 REM [INV.] FIN OE JUEGO [TRUE]
790 PRINT  PAPER 2; INK 6; FLASH 1;AT 17,2;"TE HAN F
ALTADO ";198-T;" LADRILLOS"
800 PRINT AT 19,2;"PULSA UNA TECLA PARA EMPEZAR"
810 LET K$=INKEY$
820 IF K$="" THEN  GO TO 810
830 RUN
840 REM [INV.] INTRO TOPE [TRUE]
850 CLS
860 PRINT AT 19,1;"CUANTAS VIOAS QU1ERES ?  (3-9)"
870 LET K$=INKEY$
880 IF K$="" THEN  GO TO 870
890 IF CODE K$<51 OR CODE K$>57 THEN  GO TO 870
900 BEEP .1,40
910 LET TOPE=VAL K$
920 RETURN
930 REM [INV.] PAREO OESTRUIDA [TRUE]
940 INK 6: PAPER 1: BORDER 1
945 CLS
950 PRINT  FLASH 1;AT 10,3;"  ERES UN TIO MUY HABIL!
  "
960 PRINT  OVER 1;AT 13,2;"PULSA UNA TECLA PARA EMPE
ZAR"
970 LET K$=1NKEY$
980 IF K$="" THEN  GO TO 960
990 RUN
1000 PEM [INV.] GPAFICOS DE USUAP1O [TRUE]
1010 POKE 32767,1
1020 LET A$="ABCEFG"
1030 FOR N=1 TO LEN A$
1040 FOR F=0 TO 7
1050 READ A
1060 POKE USR A$(N)+F,A
1070 NEXT F
1080 NEXT N
1090 PETURN
1100 REM [INV.] DATAS [TRUE]
1110 OATA 255,128,128,128,128,128,128,255
1120 OATA 255,1,1,1,1,1,1,255
1130 OATA 60,126,255,255,255,255,126,60
1140 OATA 127,255,255,255,127,1,3,15
1150 OATA 255,255,255,255,255,195,129,0
1160 OATA 254,255,255,255,254,128,192,240
```